USACO Gold Dynamic Programming Cheatsheet

Use this algorithm when:

- You can brute force this problem with a recursive method.
- The brute force has a lot of wasted/repeat computations.
- It's very common to use DP for problems where you need to make many decisions, or count many things.

1) Run a brute force by hand on a small, but not tiny input (could be the sample)
2) See if you did "duplicate work"
3) If you did duplicate work, it is very likely dp

Steps to using the algorithm:

Thinking about the problem:

1) Figure out the state
    a) This is often the hardest step.
    b) It's often helpful to process the input in a certain order (SORT in some order) and make your index your state.
    c) If you're working on a decision problem, each decision is a transition, and the intermediate points between each decision is your state.
    d) Adding extra **positional** state variables.
        i) It's impossible to formulate this DP problem using one position.
        ii) You NEED to add more dimensions into your DP.
        iii) 3 most common examples are:
            - DPing on a grid, instead of a list.
            - DP on two lists.
            - DP using a range of values as a state rather than a single value.
    e) Adding extra **non-positional** state variables.
        i) It is impossible to formulate this DP problem using only positional variables.
        ii) It is USEFUL to add more dimensions into your DP
        iii) 2 most common examples are:
            - Knapsack DP.
            - DP where you can make changes at K points out of N.
2) Figure out the transitions
    a) If there are decisions, each decision is a transition
3) Figure out the base case and answer state

****Ensure your program will run in time before implementing it****

Implementation:

4) Create an array (or map if needed) where the index of the array is the state. This can be a multi-dimensional array!
5) Set the base cases
6) Find an order to look at the states in based on the transitions. Loop in that order and do the transitions
7) Print out the answer state

How to calculate the runtime:

(# of states) * (# of transitions) * (any extra runtime, ie if you are using a set, this will be log(n))